Cloud Computing

Adrien Reymann

Table des matières

1. Problématique

L'algorithme doit prédire le si Tweet peut lui être proposé en fonction de son nombre de retweet sur le sujet.

2. Contrainte

Seul le texte sera passé dans l'api et celui-ci sera proposé aux utilisateurs présents dans le data si la prédiction est supérieure à 30%

3. Présentation donnée

Dataset choisis est un dataset sur les retweets sur le nombre de retweet à propos du vaccin. Celui est composé de 2207 ligne × 16 colonnes.

Les colonnes que nous utiliserons pour la prédiction seront le nombres retweet ainsi que le tweet(text).

Forme du dataset (voir les captures ci-dessous)

```
Entrée [18]: df.head()
```

Out[18]:

	id	user_name	user_location	user_description	user_created	user_followers	user_friends	user_favourites	user_verified	
0	1340539111971516416	Rachel Roh	La Crescenta-Montrose, CA	Aggregator of Asian American news; scanning di...	2009-04-08 17:52:46	405	1692	3247	False	202
1	1338158543359250433	Albert Fong	San Francisco, CA	Marketing dude, tech geek, heavy metal & '80s ...	2009-09-21 15:27:30	834	666	178	False	202
2	1337858199140118533	eli	Your Bed	heil, hydra	2020-06-25 23:30:28	10	88	155	False	202
3	1337855739918835717	Charles Adler	Vancouver, BC - Canada	Hosting "CharlesAdlerTonight" Global News Radi...	2008-09-10 11:28:53	49165	3933	21853	True	202
4	1337854064604966912	Citizen News Channel	NaN	Citizen News Channel bringing you an alternati...	2020-04-23 17:58:42	152	580	1473	False	202

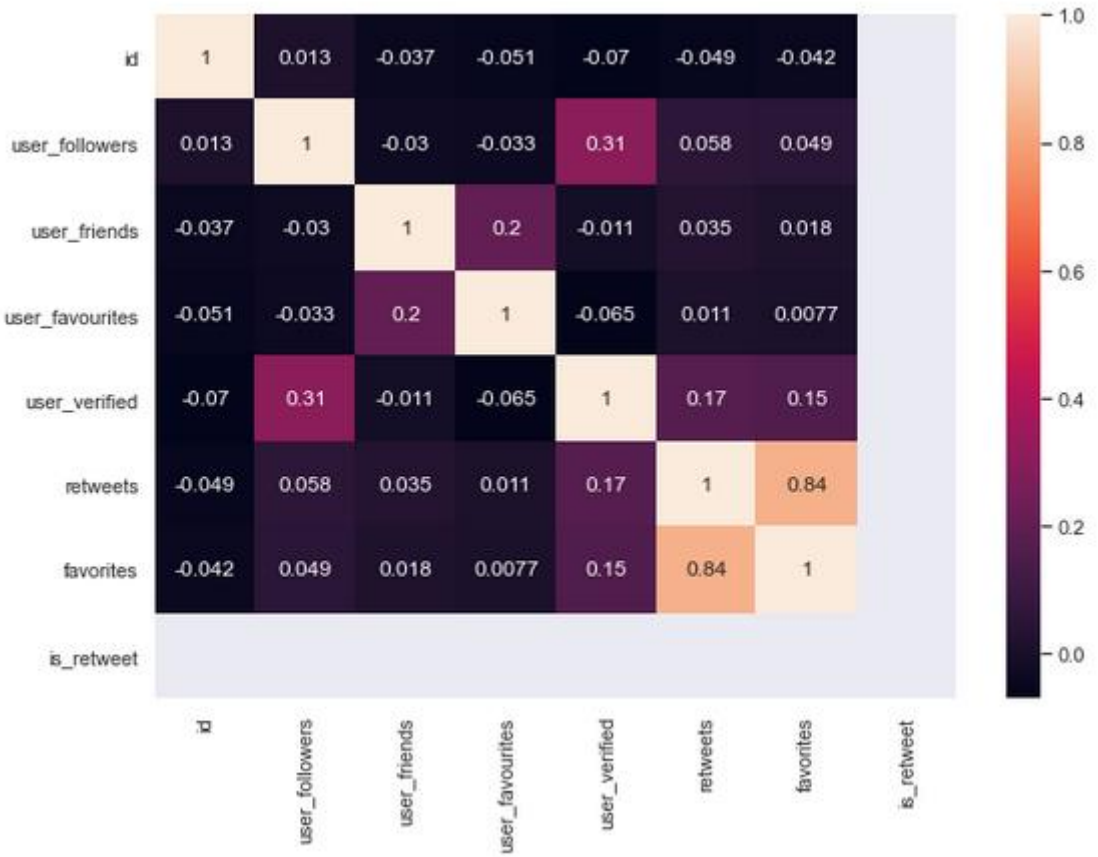
```
Entrée [18]: df.head()
```

Out[18]:

user_followers	user_friends	user_favourites	user_verified	date	text	hashtags	source	retweets	favorites	is_retweet
405	1692	3247	False	2020-12-20 06:06:44	Same folks said daikon paste could treat a cyt...	['PfizerBioNTech']	Twitter for Android	0	0	False
834	666	178	False	2020-12-13 16:27:13	While the world has been on the wrong side of ...	NaN	Twitter Web App	1	1	False
10	88	155	False	2020-12-12 20:33:45	#coronavirus #SputnikV #AstraZeneca #PfizerBio...	['coronavirus', 'SputnikV', 'AstraZeneca', 'Pf...	Twitter for Android	0	0	False
49165	3933	21853	True	2020-12-12 20:23:59	Facts are immutable, Senator, even when you're...	NaN	Twitter Web App	446	2129	False
152	580	1473	False	2020-12-12 20:17:19	Explain to me again why we need a vaccine @Bor...	['whereareallthesickpeople', 'PfizerBioNTech']	Twitter for iPhone	0	0	False

4. Corrélation

Sur la heatmap ci-dessous on peut voir qu'il y a une corrélation de 58% entre les retweets et le nombre de followers



5. Pipeline

A. DataHandler

Dans cette classe nous récupérons le dataset grâce a la méthode « get_data » ensuite nous faisons passer la dataframe dans un cleaner de texte avec une regex ce que fais la méthode « clean_text ».

```
class DataHandler:

    """
        R?cup?ration des data depuis le GCS Bucket
    """

    def __init__(self):
        self.df_vaccin = None

    def get_data(self):
        """
            R?cup?ration du
        """
        self.df_vaccin= pd.read_csv("utils/vaccination_tweets.csv",sep=",",index_col=0)
        print("data charg?s")

    def clean_text(self):
        self.df['text'] = df['text'].apply(lambda tweet: re.sub('[^A-Za-z]+', ' ', tweet.lower()))

    def get_process_data(self):
        """
            Lancement des diff?rente m?thode get_data()+goup_data()
        """
        self.get_data()
        self.clean_text()
        print('Data processed')
        return self.get_data()
```

B. FeatureRecipe

Dans cette classe nous afficher séparons les données dans un tableau par types via la méthode « separate_variable_types » une fois cela fait nous supprimons les colonnes inutiles avec la méthode « drop_uselessf » et ensuite nous allons supprimer les colonnes qui sont dupliqué via la méthode « drop_duplicate » une fois les colonnes inutile et dupliqué supprimer on va vérifier si tout les data sont bien représenté si le nombre de NaN est supérieur a 3% on regroupe celle-ci dans une colonne.

```
class FeatureRecipe:

    def __init__(self, data: pd.DataFrame()):
        self.df_data = data
        self.cate = []
        self.floa = []
        self.intt = []
        self.drop = []


    def separate_variable_types(self) -> None:
        """ TODO : Diviser les types de variables dans un tableau """
        #variables = self.df_data.dtypes
        #var_tab = Arrays('int64',['128946'])
        #print(var_tab[int64])

        """correction"""
        print("separating columns")
        for col in self.df_data.columns:
            if self.df_data[col].dtypes == int:
                self.intt.append(self.df_data[col])
            elif self.df_data[col].dtypes == float:
                self.floa.append(self.df_data[col])
            else:
                self.cate.append(self.df_data[col])
        print ("dataset column size : {} \nnumber of discreet values : {} \nnumber of continuous values : {} \nnumber of
others : {} \ntaille total :
{}".format(len(self.df_data.columns),len(self.intt),len(self.floa),len(self.cate),len(self.intt)+len(self.floa)+len(self
.cate) ))


    def drop_uselessf(self):
        """ TODO : Supprimer les colonnes inutiles du dataset """
        colonnes_drop=['user_created',
                       'user_followers',
                       'user_friends',
```

```
    def drop_duplicate(self):
        """ TODO : Supprimer les lignes dupliqu?es du dataset """
        print('duplicate')
        a=0
        for i in self.df_data:
            a+=1
            b=0
            for j in self.df_data:
                b +=1
                if a != b and self.df_data[i].equals(self.df_data[j]) == True:
                    self.df_data.drop(columns=j)
                    print('{} supprim?e'.format(j))

    def Verif_data(self):
        print('verif date')
        """ V?rif des data sup?rieur a 3%"""
        for colonne in  self.df_data:
            nbNaN =  self.df_data[colonne].isna().sum()
            if (nbNaN /  self.df_data.shape[0]) * 100 > 3:
                del  self.df_data[colonne]
                print('{} supprim?e'.format(colonne))
```

C.

```
    def prepare_data(self):
        self.separate_variable_types()
        self.drop_uselessf()
        self.drop_duplicate()
        self.Verif_data()
        return self.df_data
```

Cette classe vas créer la pipeline soit le les data qui seront prise en compte et celle-ci qui seront testé pour la précision de l'algorithme

```
class FeatureExtractor:
    """
    Feature Extractor class
    """
    def __init__(self, data: pd.DataFrame, flist: list = None):
        """
            Input : pandas.DataFrame, feature list to drop
            Output : X_train, X_test, y_train, y_test according to sklearn.model_selection.train_test_split

        """
        self.df_data = data
        self.X = self.df_data['text']
        self.y = self.df_data['retweets']
        self.clf = None

    def make_pipeline(self):
        self.clf = make_pipeline(
                            TfidfVectorizer(stop_words=get_stop_words('en')),
                            OneVsRestClassifier(SVC(kernel='linear', probability=True))
                            )

        self.clf = self.clf.fit(X=self.X, y=self.y)

        return self.clf
```

D. ModelBuild

Dans cette classe nous allons dans un premier temps entrainer celui-ci via la méthode « train » ensuite nous allons tester une prediction avec la méthode « predict_test » ensuite nous allons afficher la précision avec la méthode « print_accuracy » et enfin sauvegarder le model avec la méthode « save_model ».

```
class ModelBuild:
    def __init__(self, model_path, save, n_estimators):
        """
        constructeur
        """
        self.model_filename = model_path
        self.saveModel = save
        self.date = date.today().isoformat()
        self.n_estimators=n_estimators

    def train(self,clf,X,Y):
        clf.fit(X, Y)

    def predict_test(self,clf,text):
        #test
        clf.predict_proba([text])[0]

    def print_accuracy(self,clf,X,Y):
        """
            affichage de la precision des predictions
        """
        accuracy=clf.score(X,Y)
        print('precision : {}'.format(accuracy))

    def FeatureImportance(self,X,Y,clf):
        """
            attribut un score aux valeurs utilis? pour la prediction bas?
            sur leurs utilit?
        """
        clf.fit(X,Y)
        importance =  clf.coef_()
        for i,v in enumerate(importance):
            print('Feature: %0d, Score: %.5f' % (i,v))
        pyplot.bar([x for x in range(len(importance))], importance)
        pyplot.show()

    def save_model(self,clf):
        #save weights
        model_filename = "model.joblib.z"
        joblib.dump((clf), model_filename)

    def calculData(self,clf,X,Y,text):
        self.train(clf,X,Y)
        self.predict_test(clf,text)
        self.print_accuracy(clf,X,Y)
        #self.FeatureImportance(X,Y,clf)
        self.save_model(clf)
```

6. Api

L'Api de l'algorithme sera déployée sur un docker et ensuite mise en ligne sur heroku il y aura un formulaire demandant le texte à prédire et il affichera la précision de la prédiction.

7. Conclusion

En Conclusion l'algorithme est considéré comme overfitting car la regex n'est pas assez précise dans la suppression des caractères et il n'y a pas assez de donnée d'entrainement pour avoir une prédiction assez précise et correct.